CM # XX-XXX-XXX-XX: Vol. 1 of 1

Revision 2.0: 11 July 1997

DRAFT

SOFTWARE REQUIREMENTS SPECIFICATION

FOR THE

TOOLKIT FUNCTIONAL AREA

OF THE

DEFENSE INFORMATION INFRASTRUCTURE
COMMON OPERATING ENVIRONMENT (DII COE)

Prepared for and by:

Defense Information System Agency (DISA)
Joint Interoperability and Engineering Organization (JIEO)
DII COE Engineering Office (JEXF)
Operational Support Facility
45335 Vintage Park Plaza
Sterling VA 20166-6701

Approved by _____
Date _____

Signature Sheet

The Software Requirements Specification (SRS) for the Toolkit functional area of the Defense Information Infrastructure Common Operating Environment (DII COE), Revision 2.0, 11 July 1997:

SUBMITTED BY:

_____

Daniel Test,
Chairperson,
DII COE Toolkit Technical Working Group (TKTWG)
Joint Interoperability and Engineering Organization
Defense Information Systems Agency

APPROVED BY:

_____

Dawn A. Hartley,
DII COE Chief Engineer,
Center for Computer Systems Engineering
Joint Interoperability and Engineering Organization
Defense Information Systems Agency

Comments should be electronically mailed to the Chair, Mr. Daniel Test at testd@ncr.disa.mil (703) 735-8736. New requirements or suggested changes should be sent to the reviewer's Service or Agency Architecture Oversight Group (AOG) representative who inturn will review/deconflict them and forward, as appropriate, to the Toolkit TWG Chair.

Table of Contents

Section **Page**

1. SCOPE

1.1 Identification

This Software Requirements Specification (SRS) describes the requirements for Toolkit (TK) functional area of the Defense Information Infrastructure (DII) Common Operating Environment (COE). The Toolkit Services functional area falls under the overall Kernel functional area. This SRS is limited to the Developer's Toolkit and other development and support toolkit functional concerns of the DII COE.

The SRS is independent of a particular DII COE version. Instead, this document contains the total objective set of DII COE toolkit requirements that DISA, as the Executive Agent (EA) of the DII COE, strives to fulfill. This document only applies to those Developer's Toolkit applications and libraries, utility development toolkit applications and libraries, that are considered an integral part of the DII COE development environment and fall under the direct control and supervision of the DII COE Engineering Office (DISA/JEXF-OSF). There may be cases where supporting development applications are in use by DII COE-compliant development contractor's and Government personnel but they are not considered part of the DII COE because they are not under the DII COE Engineering Office's direct control and supervision. This unique set of supporting software development tools and libraries are considered for use internal to a specific community of interest. They fall outside the scope of this SRS.

Although other DII COE functional areas support their own unique software development tools and libraries, they fall outside the detailed Scope of this SRS. Those additional tools, toolkits, and libraries, and the functional area they belong to will be identified in this SRS, along with associated requirements for common look and feel format, implementation and operation, supporting documentation, and distribution.

Additional software development requirements (requirements that a developer must comply with in order to be DII COE compliant) are described in the Integration and Runtime Specification (I&RTS), Style Guide, and other DII COE documents.

1.2 System Overview

The DII COE concept is best described as an architecture that is fully compliant with the DOD Technical Architecture for Information Management (TAFIM), Volume 3, an approach for building interoperable systems, a reference implementation containing a collection of reusable software components, a software infrastructure for supporting mission-area applications, and a set of guidelines, standards, and specifications. The guidelines, standards, and specifications describe how to reuse existing software and how to properly build new software so that integration is seamless and, to a large extent, automated. The DII COE will evolve as necessary to become compliant with emerging specifications, such as the Joint Technical Architecture (JTA) and TAFIM. The JTA stipulates DII compliance as part of its requirements and replaces the standards guidance in the TAFIM as per an OSD directive dated 30 August 1996.

The objective of the DII COE Toolkit TWG is to provide the developer community with the assets needed to develop software for the DII repository. The Developer's Toolkit will contain software tools, documentation, and other information items used to develop DII compliant software. The Toolkit TWG has two primary areas of focus which are presented in the Toolkit SRS: Runtime Tools and the Developer's Toolkit.

The Runtime Tools are packaged as part of the DII COE kernel. They are available from the command line and include tools that are necessary for user Interface, database initialization, system administration, profiling, and segmentation.

The Developer's Toolkit is intended to provide a single source packaging of the API-related files from those other segments (along with tools, documentation, and other information). The single source packaging enables a quicker ramp-up than would be the case if these files had to be collected by each individual developer on a segment-by-segment basis.

The Developer's Toolkit contains the components necessary for creating segments that use COE components. The Toolkit does not need to be in segment format (it is not installed at operational sites), but it is a set of files and directories that may be downloaded electronically from the on-line library. Developer's may also contact the DII COE Configuration Management Department to receive the Toolkit on magnetic media in relative "tar" format. The Developer's Toolkit is distributed separately from the target COE-based system. However, components from the operational system (COE-component segments, shared libraries, etc.) are required for development. These may be obtained electronically from the on-line library, or on magnetic media from the DII COE Configuration Management Department. Classified or very large components will be distributed to developers via magnetic media. The Toolkit does not duplicate any components available in the runtime system because this would create configuration management problems in ensuring that developers do not receive two different versions of the same module.

The Developer's Toolkit, as available from DII COE CM, will focus on APIs, segmentation tools, run-time integration, and compliance metrics. It will gradually increase its focus to include CASE tools, static code checking, coding standards, security rules, reengineering tools, and other items relating to new software or to software reengineering to comply with the DII COE objective architecture.

1.3 Document Overview

This document outlines the software capabilities required for the DII COE Toolkit in accordance with the content and format guidance of Data Item Description (DID) DI-IPSC-81433.

- Section 1 identifies the scope and provides an overview of the DII COE Toolkit SRS.
- Section 2 lists the documents which are applicable to and referenced in this document.
- Section 3 provides a list of functional capability requirements.
- Section 4 identifies the qualification provisions.
- Section 5 provides a requirements traceability matrix (RTM) to accompany the Toolkit requirement statements in Section 3.2.
- Section 6 contains the applicable notes associated with the DII COE Toolkit SRS.

2. REFERENCED DOCUMENTS

The following specifications, standards, and handbooks form a part of this document to the extent specified herein. Unless otherwise specified, the issues of DII COE documents are available on the Internet under the DII COE Home Page under CM Documentation.

2.1 Government Documents

- Defense Information Infrastructure Common Operating Environment Integration and Runtime Specification, Version 3.0 (Draft), January 1997
- Department of Defense Technical Architecture Framework for Information Management, Volumes 1-8, Version 3.0 (Draft), 30 September 1995.

3. REQUIREMENTS

3.1 Required States and Modes

The tools shall have the following command-line parameters which are common to all tools for which they are meaningful:

-C file Read command-line parameters from file.
-h, H Display on-line help describing how to use the tool.
-p path Use path to establish the path for subsequent file names.
-R file Use file to respond to questions from the tool.
-v Toggle the "verbose" flag. When enabled, print out diagnostic information as the tool executes. The initial state of the flag is disabled.
-V Display the tool's version number.
-w Toggle the "warnings" flag. When enabled, print out warning messages as the tool executes. The initial state of this flag is enabled.

3.1.1 General Requirements

3.1.1.1 The tools specified in this SRS shall output textual (help and version number) information to *stdout* and errors to *stderr*.

3.1.1.2 The operational behavior of the tools specified in this SRS shall be the same whether invoked as a standalone executable, or invoked as a subroutine in a library.

3.2 Toolkit Functional Area Capability Requirements

3.2.1 Runtime Tools

3.2.1.1 User Interface

3.2.1.1.1 The runtime tools shall provide a *COE prompt* tool with the capability to display a message to the user.

3.2.1.1.1.1 The tool shall allow the user to type in a response.

3.2.1.1.1.2 The maximum number of characters allowed in the response shall be capable of being set by the calling routine.

3.2.1.1.1.3 The default maximum number of characters allowed in the response shall be 40 characters.

3.2.1.1.2 The runtime tools shall provide a *COE ask user* tool with the capability to ask a user for a “yes” or “no” response to a message.

3.2.1.1.2.1 The tool shall include the capability to define the message to ask the user.

3.2.1.1.2.2 The tool shall be executed using the following syntax:

```
FunctionName {parameters} msg
```

where `msg` is the prompt to display to the user.

3.2.1.1.2.3 The tool shall use the `parameters` input to specify the labels to be used for “yes/no” buttons.

3.2.1.1.3 The runtime tools shall provide a *COE message* tool with the capability to display a message to the user.

3.2.1.1.3.1 The tool shall be able to be used in within `PreInstall`, `PostInstall`, and `DEINSTALL` scripts.

3.2.1.1.3.2 The user shall be required to respond to an “OK” button to continue.

3.2.1.1.4 The runtime tools shall provide a *COE prompt/password* tool with the capability to prompt a user for a password.

3.2.1.1.4.1 The tool shall be able to be used in within `PreInstall` and `PostInstall` scripts.

3.2.1.1.4.2 The user’s password response shall not be viewable.

3.2.1.1.4.3 By default, the runtime tool shall prompt for the password, then prompt the user to enter the password again for confirmation.

3.2.1.1.4.4 The runtime tool shall provide for the capability to have a flag passed to the tool by a calling routine to indicate that a confirmation is not required.

3.2.1.1.4.5 The default maximum password length shall be 14 characters and the minimum shall be 6 characters.

3.2.1.1.4.6 The runtime tool shall be able to have a parameter passed to it to set the maximum and minimum number of characters in the password.

3.2.1.1.4.7 When a calling routine requests confirmation, the runtime tool shall prompt the user for a matching confirmation up to 3 times before returning a failure.

3.2.1.1.4.8 If after 3 attempts the confirmation does not match, the runtime tool shall return an error code to the calling routine.

3.2.1.2 Database Tools

3.2.1.2.1 The runtime tools shall provide a *COE add database user* tool with the capability to add database server user accounts.

3.2.1.2.1.1 The users' permissions associated with their database user account shall be synchronized with the permissions associated with their application profiles.

3.2.1.2.1.2 The tool shall use the database role information under the application segment's `Database` descriptor in `SegInfo` to do this.

3.2.1.2.1.3 The tool shall be automatically accessed when accounts are added with profiles that include database access.

3.2.1.2.1.4 The tool shall also be capable of being accessed independently as part of a database administrator account group.

3.2.1.2.2 The runtime tools shall provide a *COE delete database user* tool with the capability to delete database server user accounts.

3.2.1.2.2.1 The users' permissions associated with their database user account shall be synchronized with the permissions associated with their application profiles.

3.2.1.2.2.2 The tool shall use the database role information under the application segment's `Database` descriptor in `SegInfo` to do this.

3.2.1.2.2.3 The tool shall be automatically accessed when accounts are deleted with profiles that include database access.

3.2.1.2.2.4 The tool shall also be capable of being accessed independently as part of a database administrator account group.

3.2.1.2.3 The runtime tools shall provide a *COE database backup* tool with the capability to perform standard dump and restore functions on a COE database.

3.2.1.2.3.1 The tool's special format data backup options shall also be provided as specified by data store segments.

3.2.1.2.3.2 The tool's feature shall be available from a database administrator account group.

3.2.1.2.4 The runtime tools shall provide a *COE list database dependencies* tool with the capability to search object dependencies across multiple database segments.

3.2.1.2.4.1 The tool shall include optional input parameters to search for all dependencies of a specified data object.

3.2.1.2.4.2 The tool shall also include optional input parameters to search for all dependencies of a database owner's schema on other schema's objects.

3.2.1.2.5 The runtime tools shall provide a *COE list user accounts per database role* tool with the capability to provide a sorted list of all user's database login accounts assigned to a specific database role.

3.2.1.2.5.1 The tool shall be available from a database administrator account group.

3.2.1.2.6 The runtime tools shall provide a *COE list database roles per user* tool with the capability to provide a sorted list of all database roles assigned to a specific user.

3.2.1.2.6.1 The tool shall be available from a database administrator account group.

3.2.1.2.7 The runtime tools shall provide a *COE restore database* tool with the capability to restore backup data stores including data and structure.

3.2.1.2.7.1 The tool shall also provide the capability to run the restore scripts for other data segments to complete the restore process.

3.2.1.2.7.2 The tool shall provide the capability for the owning segment to tag the Restore script with ".main".

3.2.1.2.7.3 The tool shall run the "`.main`" script first, followed by other public data stores, and then private data stores which have `Restore` scripts for the data store being stored.

3.2.1.2.8 The runtime tools shall provide a *COE start database server* tool with the capability to start a specified database server if server is not running.

3.2.1.2.8.1 The tool shall start the database server in its maintenance mode.

3.2.1.2.9 The runtime tools shall provide a *COE stop database server* tool with the capability to shutdown a database server to allow installation of a database segment.

3.2.1.2.9.1 The tool shall provide the capability to locate the DBMS server to be shutdown by using the supplied name provided by the interfaces and hosts file.

3.2.1.2.9.2 The tool shall use the DBMS' shutdown commands for the normal shutdown of a DBMS server.

3.2.1.2.10 The runtime tools shall provide a *COE create data store* tool with the capability to create data stores on a COE database server.

3.2.1.2.10.1 The tool shall allocate physical storage to the data stores.

3.2.1.2.10.2 This tool shall create the database owner account associated with the data stores.

3.2.1.2.10.3 The tool shall also provide the capability to be used to create a new data store for an existing database owner.

3.2.1.2.10.4 Should the tool encounter an error, it shall clean up any files, accounts, or other objects what were created during processing.

3.2.1.2.11 The runtime tools shall provide a *COE drop data store* tool with the capability to remove the files associated with an existing data store from the database server.

3.2.1.2.11.1 The tool shall be run after all data objects have been removed with a `DEINSTALL` script.

3.2.1.2.11.2 If there are any remaining data objects left in the data store to be remove, the tool shall cause a failure of the removal process.

3.2.1.2.11.3 If neither a store name nor a store list is provided in the tool's invocation, the tool shall attempt to remove all data stores associated with the specified database owner (DBO).

3.2.1.2.11.4 If neither a store name nor a store list is provided in the tool's invocation, the tool shall also remove the DBO account from the DBMS.

3.2.1.2.11.5 Should the tool encounter an error, the tool shall restore any files, accounts, or other objects that were deleted during processing.

3.2.1.2.12 The runtime tools shall provide a *COE extend data store* tool with the capability to add additional physical storage to data stores.

3.2.1.2.12.1 The tool shall require that a database owner account and the data store name exist before extending the data store.

3.2.1.2.12.2 Should the tool encounter an error during processing, it shall clean up any files or other objects that were created before the error occurred.

3.2.1.2.12.3 Should the tool encounter an error during processing, the associated application programmer interface (API) shall return an error status to the calling segment installation script.

3.2.1.2.13 The runtime tools shall provide a *COE rebuild index* tool with the capability to recover or restore corrupted indexes within a database server.

3.2.1.2.13.1 The tool shall recreate the specified index.

3.2.1.2.13.2 The tool shall also be available from a database administrator account group.

3.2.1.2.14 The runtime tools shall provide a *COE rebuild view* tool with the capability to recover or restore corrupted views within a database server.

3.2.1.2.14.1 The tool shall save any existing grants to users or database roles.

3.2.1.2.14.2 The tool shall recreate the specified view.

3.2.1.2.14.3 The tool shall re-execute the saved grants to finish restoring the view.

3.2.1.2.14.4 The tool shall be available from a database administrator account group.

3.2.1.3 System Administration Tools

3.2.1.3.1 The runtime tools shall provide a *COE check updates* tool with the capability to check for the availability of new segments, segment updates or patches.

3.2.1.3.1.1 The tool shall be capable of being invoked from the system administration account group.

3.2.1.3.1.2 The tool shall allow for the specification of a master machine.

3.2.1.3.1.3 The tool shall check the master machine for the availability of new segments.

3.2.1.3.1.4 The tool shall check the master machine for the availability of segment updates.

3.2.1.3.1.5 The tool shall check the master machine for the availability of segment patches.

3.2.1.3.1.6 The tool shall allow a member of the system administration account group to check the master server on demand.

3.2.1.3.1.7 The tool shall allow a member of the system administration account group to check the master server at specified intervals (e.g., every 24 hours).

3.2.1.3.1.8 The tool shall provide a default of checking on demand.

3.2.1.3.1.9 When the tool determines that system updates are available, it shall display a notification message to the logged in operator that new features are available.

3.2.1.3.1.10 When the system administrator receives the notification and logs in, the tool shall display a list of available new features for downloading.

3.2.1.3.2 The runtime tools shall provide a *COE connect application server* tool with the capability to connect a client workstation to an application server.

3.2.1.3.2.1 The tool shall be capable of being selected from the system administration account group.

3.2.1.3.2.2 The tool shall require, at a minimum, the client workstation be configured with the COE kernel installed.

3.2.1.3.2.3 The tool shall, by default, not load (install) segments from the application server onto the client workstation's hard disk. As required, applications are downloaded from the server and installed into the workstation's memory for execution.

3.2.1.3.2.4 The tool shall provide an option feature that allows dynamic loading of segments from the application server to the client workstation's hard disk the first time an operator attempts to access a function in the segment.

3.2.1.3.2.5 The tool shall remove the dynamically loaded segment from the client workstation when the operator logs out.

3.2.1.3.2.6 The tool shall perform a check each time a new operator logs in to a "dynamic" workstation to ensure that segments are not inadvertently left on the client workstation.

3.2.1.3.3 The runtime tools shall provide a *COE copy workstation* tool with the capability to allow segments on one workstation, the source workstation, to be copied across the network onto another workstation, the target workstation.

3.2.1.3.3.1 The tool shall configure the target workstation identically to the source workstation, including ensuring that the COE kernel is installed and configured.

3.2.1.3.3.2 The tool shall not modify workstation-specific items such as the target workstation's hostname and IP address.

3.2.1.3.3.3 The tool shall affect existing disk partitions or other actions normally performed when the bootstrap COE has been loaded on the target workstation.

3.2.1.3.3.4 The tool shall check the target workstation to ensure it has sufficient disk space to hold all of the segments that are to be transferred.

3.2.1.3.3.5 The tool shall report an error message and aborts the copy process if there is not enough storage on the target workstation.

3.2.1.3.4 The runtime tools shall provide a *COE create application server* tool with the capability to allow a site administrator to create an application server.

3.2.1.3.4.1 The tool shall be selected from the system administration account group.

3.2.1.3.4.2 The tool shall provide the capability to load segments onto a server's hard disk which, when requested by a user, gets downloaded to a client workstation for execution.

3.2.1.3.4.3 The tool shall allow segments for multiple hardware types to be loaded on the application server.

3.2.1.3.4.4 The tool shall ensure multiple versions of the same segment do not exist on the application server.

3.2.1.3.5 The runtime tools shall provide a *COE delete unused segment* tool with the capability to examine all of the installed segments and determine which are unused (e.g. not in any profile, no other segment is dependent upon it).

3.2.1.3.5.1 The tool shall generate a list a list of unused segments.

3.2.1.3.5.2 The tool shall present this list to the system administrator with the option of deleting one or more of the segments.

3.2.1.3.5.3 The tool shall be able to be selected from a system administrator's menu.

3.2.1.3.5.4 The tool shall also be executed from the command line.

3.2.1.3.5.5 The tool shall be able to be executed from one workstation to examine itself or another workstation.

3.2.1.3.6 The runtime tools shall provide a *COE find data segment* tool with the capability to determine where a data segment was loaded.

3.2.1.3.6.1 The tool shall be invoked from a `PostInstall` or other installation-related script.

3.2.1.3.7 The runtime tools shall provide a *COE find server* tool with the capability to determine the identity of a requested server.

3.2.1.3.7.1 The tool shall provide the requested server's hostname.

3.2.1.3.7.2 The tool shall provide the requested server's IP address.

3.2.1.3.7.3 The tool shall provide whether or not the server is a Distributed Computing Environment (DCE) server.

3.2.1.3.7.4 The tool shall use the server name specified by the $SERVERS or $DCESERVERS keyword in the Network descriptor.

3.2.1.3.7.5 If the server is a DCE server, the tool shall return the hostname and IP address of the first host where the server is found if there are multiple hosts running the requested server.

3.2.1.3.8 The runtime tools shall provide a *COE list segment* tool with the capability to display a sorted list of all segments that have been installed on a specific machine.

3.2.1.3.8.1 The tool shall be selected from the system administration menu.

3.2.1.3.8.2 The tool shall be executed from the command line with output written to `stdout`.

3.2.1.3.8.3 The tool shall also be executed through a graphical user interface (GUI).

3.2.1.3.9 The runtime tools shall provide a *COE list segment dependencies* tool with the capability to display a sorted list of all segments upon which a specified segments depends.

3.2.1.3.9.1 The tool shall provide the capability to be selected from a system administrator menu.

3.2.1.3.9.2 The tool shall also provide the capability to be executed from a command line.

3.2.1.3.9.3 If no segment name is provided as an input parameter, the tool shall display a GUI for the user to interact with.

3.2.1.3.9.4 The tool shall generate a list of segments accessible on the machine to be displayed in the GUI for the user to select from.

3.2.1.3.9.5 Once the user selects the segment from the segment access list, the GUI shall display the sorted list of all dependent segments.

3.2.1.3.9.6 If a segment name is provided as an input parameter, the tool shall assume the tool is being run from a command line and the tool writes the sorted list of dependent segments to `stdout`.

3.2.1.3.10 The runtime tools shall provide a *COE propagate updates* tool with the capability to allow segments, upgrades and patches to be sent to other workstations.

3.2.1.3.10.1 The tool shall be accessible only to the system administrator.

3.2.1.3.10.2 The tool shall determine which workstations need to be upgraded.

3.2.1.3.10.3 Once determined which workstations need to be upgraded, the tool shall automatically perform the upgrades.

3.2.1.3.11 The runtime tools shall provide a *COE remove application server* tool with the capability to remove a segment from an application server.

3.2.1.3.11.1 The tool shall be selectable from the system administration account group.

3.2.1.3.11.2 The tool shall, upon detecting all segments in a particular hardware type are removed, shall remove all directories associated with the hardware type on the application server.

3.2.1.3.12 The runtime tools shall provide a *COE test segment installation* tool with the capability to allow a site administrator to temporarily install a segment on an isolated test workstation for the purposes of checking out the segment without loading it from tape or other electronic media.

3.2.1.3.12.1 The tool shall be selectable from the system administration account group.

3.2.1.3.12.2 The tool shall be accessible from a command line.

3.2.1.3.12.3 The tool shall be capable of temporarily installing an individual segment.

3.2.1.3.12.4 The tool shall be capable of temporarily installing a collection of segments from a configuration definition.

3.2.1.3.12.5 When a segment is temporarily installed, the tool shall be capable of accepting an expiration date from the user.

3.2.1.3.12.6 The tool shall provide a default of 48 hours as the expiration date.

3.2.1.3.12.7 When the expiration date arrives, the tool shall display a warning message to the operator upon login to indicate that a segment has expired.

3.2.1.3.12.8 If the test is successful, the tool shall provide the capability to promote the segment from “test” status to “installed” without the need to delete and reinstall the segment.

3.2.1.4 Profiling Tools

3.2.1.4.1 The runtime tools shall provide a *COE list segments per profile* tool with the capability to display a sorted list of all segments accessible from a specified profile.

3.2.1.4.1.1 The tool shall be selectable from the system administration account group.

3.2.1.4.1.2 The tool shall be accessible from a command line.

3.2.1.4.1.3 If no profile is specified, the tool shall provide a GUI for the user to interact with.

3.2.1.4.1.4 The tool shall generate a list of profiles to be displayed in the GUI for the user to select from.

3.2.1.4.1.5 Once the profile has been selected in the GUI, the tool shall generate and display a list of all segments accessible from that profile.

3.2.1.4.1.6 If a profile is specified as an input parameter, the tool shall assume the tool is being run from a command line and writes the sorted list of segments accessible from the specified profile to `stdout`.

3.2.1.4.2 The runtime tools shall provide a *COE list user account per profile* tool with the capability to display a sorted list of all user login accounts that are assigned to a specific profile.

3.2.1.4.2.1 The tool shall be selectable from the system administration menu.

3.2.1.4.2.2 The tool shall be accessible from a command line.

3.2.1.4.2.3 The tool shall provide the capability to display a sorted list of all local user login accounts that are assigned to a local profile.

3.2.1.4.2.4 The tool shall provide the capability to display a sorted list of all global user login accounts that are assigned to a global profile.

3.2.1.4.2.5 If no profile is specified, the tool shall prompt the user to select a profile from a list of profiles available on the machine.

3.2.1.4.2.6 If a profile is specified, the tool shall assume that execution was launched from the command line and write the output to `stdout`.

3.2.1.4.3 The runtime tools shall provide a *COE list profile containing segment* tool with the capability to display a list of all profiles sorted by account group that contain a specified segment.

3.2.1.4.3.1 The tool shall be selectable from the system administration menu.

3.2.1.4.3.2 The tool shall be accessible from a command line.

3.2.1.4.3.3 The tool shall provide the capability to display a list of segments accessible from the machine.

3.2.1.4.3.4 If no segment name is specified, the tool shall prompt the user to select a segment from the list of segments accessible from the machine.

3.2.1.4.3.5 If a segment name is specified, the tool shall assume that execution was launched from the command line and write the output to `stdout`.

3.2.1.4.3.6 Once the segment name is specified, the tool shall generate and display a list of all the profiles, that contain that segment, sorted by account group.

3.2.1.4.4 The runtime tools shall provide a *COE list user account with segment access* tool with the capability to display a sorted list of all user login accounts that have access to a given segment.

3.2.1.4.4.1 The tool shall be selectable from the system administration menu.

3.2.1.4.4.2 The tool shall be accessible from a command line.

3.2.1.4.4.3 The tool shall provide the capability to display a sorted list of all local user login accounts that have access to a given segment.

3.2.1.4.4.4 The tool shall provide the capability to display a sorted list of all global user login accounts that have access to a given segment.

3.2.1.4.4.5 If no segment name is specified, the tool shall prompt the user to select a segment from a list of segment accessible from the machine.

3.2.1.4.4.6 If a segment name is specified, the tool shall assume that execution was launched from the command line and write the output to `stdout`.

3.2.1.4.5 The runtime tools shall provide a *COE list user profiles* tool with the capability to display a list of all profiles sorted by account group that are assigned to a specific user.

3.2.1.4.5.1 The tool shall be selectable from the system administration menu.

3.2.1.4.5.2 The tool shall be accessible from a command line.

3.2.1.4.5.3 If no user login account name is specified, the tool shall prompt the user to select a login account from a list of local and global accounts accessible from the machine.

3.2.1.4.5.4 If a user login account name is specified, the tool shall assume that execution was launched from the command line and write the output to `stdout`.

3.2.1.4.5.5 Once the user login account name is specified, the tool shall generate and display a list of all the profiles, that are assigned to a specific user, sorted by account group.

3.2.1.4.6 The runtime tools shall provide a *COE list segments accessed by user* tool with the capability to display a sorted list of all segments that are accessible by a specific user.

3.2.1.4.6.1 The tool shall be selectable from the system administration menu.

3.2.1.4.6.2 The tool shall be accessible from a command line.

3.2.1.4.6.3 If no user login account name is specified, the tool shall prompt the user to select a login account from a list of local and global accounts accessible from the machine.

3.2.1.4.6.4 If a user login account name is specified, the tool shall assume that execution was launched from the command line and write the output to `stdout`.

3.2.1.4.6.5 Once the user login account name is specified, the tool shall generate and display a sorted list of all the segments that are accessible by a specific user.

3.2.1.5 Installation Tools

3.2.1.5.1 The runtime tools shall provide a *COE segment installation* tool with the capability to display a list of configuration definitions or segments that may be installed from tape, disk, CD-ROM, DVD, or other electronic media, and then install selected segments or configuration definitions.

3.2.1.5.1.1 The tool shall provide the capability to be executed from a system administrator's menu by an operator who has system administration privileges.

3.2.1.5.1.2 The tool shall provide the capability to be executed directly from the command line.

3.2.1.5.1.3 If the tool is executed from the command line, it shall provide the default capability of not writing any output to `stdout`, unless the `-v` parameter (verbose) and/or `-w` parameter (warning) are used.

3.2.1.5.1.4 Once the segment(s) or configuration definition(s) have been selected, the tool shall provide the capability to install those selections.

3.2.1.5.1.5 The tool shall be able to detect when the COE display installation error tool has been called.

3.2.1.5.1.6 When the segment installation tool has detected that the COE display installation error tool was called, it shall terminate the segment's installation process.

3.2.1.5.1.7 When the segment installation tool has detected that the COE display installation error tool was called, it shall display an appropriate error message.

3.2.1.5.1.8 The tool shall provide the capability to write information to a status log that indicates installation progress, which segments have been installed, and other information that might be useful to the site administrator.

3.2.1.5.1.9 The tool shall provide the capability for the site administrator user to choose to have the information generated from the `PreInstall`, `PostInstall`, and `DEINSTALL` scripts written to the status log.

3.2.1.5.1.10 The tool shall provide the capability to print the status log.

3.2.1.5.2 The runtime tools shall provide a *COE display installation error* tool with the capability to allow a segment to display an error to the user from within a `PreInstall`, `PostInstall` or `DEINSTALL` script and terminate installation of the segment.

3.2.1.5.2.1 The tool shall always return the same return code.

3.2.1.5.2.2 The tool shall cause the *COE segment installation, test segment installation,* and *test segment remove* tools to halt further processing of the segment.

3.2.1.5.2.3 Invocation of the *COE display installation error* tool from within the `PreInstall` or `PostInstall` scripts shall cause the installation tools to remove the segment because the installation was unsuccessful.

3.2.1.5.2.4 Invocation of the *COE display installation error* tool from within the `DEINSTALL` script shall cause the installation tools not to remove the segment.

3.2.1.5.3 The runtime tools shall provide a *COE remote installation* tool with the capability to remotely install one or more segments from the software repository in either a “push” or “pull” mode.

3.2.1.5.3.1 The tool shall provide the capability to be executed from a system administrator’s menu.

3.2.1.5.3.2 The tool shall provide the capability to be executed directly from the command line.

3.2.1.5.3.3 In the “push” mode (segments are sent without request), the tool shall provide the capability to send one or more segments to a remote network’s installation server.

3.2.1.5.3.4 After the segment or segments have been “pushed”, the tool shall provide the capability to automatically invoke the *COE segment installation* tool on the remote machine to allow the sending site operator to perform an actual installation.

3.2.1.5.3.5 In the “pull” mode (remote site requests segments), the tool shall provide the capability to remotely initiate the transmission of one or more segment from the sending (repository) site to be received by the remote site’s installation server.

3.2.1.5.3.6 After the segment or segments have been “pulled”, the tool shall provide the remote operator the capability to install the segment(s) on one or more machines.

3.2.1.5.3.7 The tool shall provide the capability for an operator at a sending (repository) site to remotely remove a segment on a machine at the remote site by launching *the COE segment installation* tool on the remote machine and using it to select the segment(s) to delete.

3.2.1.5.3.8 The tool shall provide security measures (encryption, passwords, anonymous ftp, digital signature, MD5, etc.) to protect the segment(s) during transmission.

3.2.1.5.3.9 The tool shall provide security measures (encryption, passwords, etc.) to prevent unauthorized access to the repository or a remote site.

3.2.1.5.4 The runtime tools shall provide a *COE scan for COTS* tool with the capability to scan the disk to find applications that are already installed and automatically create segment descriptors for them. (NT only)

3.2.1.5.4.1 The tool shall use the created segment descriptors to build a table to be used by the *COE segment installation* tool to use by GOTS segments to express dependencies on the pre-loaded COTS products.

3.2.1.5.5 The runtime tools shall provide a *COE test segment remove* tool with the capability to remove a segment that was temporarily installed.

3.2.1.5.5.1 The tool shall provide the capability to be executed from a system administrator's account group.

3.2.1.5.5.2 The tool shall only list temporarily installed segments.

3.2.1.5.5.3 The tool shall remove temporarily installed non-expired segments.

3.2.1.5.5.4 The tool shall remove temporarily installed expired segments.

3.2.1.5.5.5 The tool shall not permit the removal of any other type of segment.

3.2.1.5.5.6 The tool shall provide the operator with the option of either removing the segment, or making the installation "permanent."

3.2.1.5.5.7 When a temporary segment is removed, the tool shall provide the capability for the segment it replaced, if any, to be restored.

3.2.1.5.6 The runtime tools shall provide a *COE update home environment* tool with the capability to update the home environment variable within a script file to point to where a segment was actually installed.

3.2.1.5.6.1 The tool shall be capable of being invoked from within a segment's `PostInstall` script.

3.2.1.5.6.2 The tool shall search the named script file for the first place that the environment variable is defined through either `set` or `setenv` command.

3.2.1.5.6.3 Once found with either the `set` or `setenv` command, the tool shall replace the home environment variable with the absolute pathname for where the segment was loaded.

3.2.1.5.7 The runtime tools shall provide a *COE find drive* tool with the capability to return the name of the disk drive, which may be a network drive, that contains the specified segment. (NT only)

3.2.1.5.8 The runtime tools shall provide a *COE find segment* tool with the capability to return information about a requested segment.

3.2.1.5.8.1 The tool shall accept as input the name of the directory where the segment is expected to be, the segment name, the segment prefix, and wildcarding.

3.2.1.5.8.2 If the directory is omitted, the tool shall perform a search for the segment in the legal directory location for segments (e.g., `/h, /h/AcctGrps, /h/COE/Comp`).

3.2.1.5.8.3 If the inputted directory name is not found, the tool shall perform a search for the segment in the legal directory locations for segments (e.g., `/h, /h/AcctGrps, /h/COE/Comp`).

3.2.1.5.8.4 The tool shall return the absolute path of the directory where the segment was found.

3.2.1.5.8.5 The tool shall return the segment name found.

3.2.1.5.8.6 The tool shall return the segment prefix found.

3.2.1.5.8.7 The tool shall return the segment type, including the segment attribute if applicable.

3.2.1.5.8.8 If the tool can't find information about the requested segment, it shall notify the user by returning an error value.

3.2.2 Developer's Toolkit

3.2.2.1 Developer's Tools

3.2.2.1.1 Database Tools

3.2.2.1.1.1 The developer's tools shall provide a *COE check database compliance* tool with the capability to examine a database segment to determine its level of compliance.

3.2.2.1.1.1.1 This tool shall be called from the *COE check compliance* tool.

3.2.2.1.1.1.2 Compliance items that can not be verified via software shall be identified to allow for manual verification.

3.2.2.1.1.2 The developer's tools shall provide a *COE verify segment database* tool with the capability to validate that a database segment's database conforms to the rules for defining a segment's database.

3.2.2.1.1.2.1 This tool shall be called from the *COE verify segment* tool.

3.2.2.1.1.2.2 The data base segment verification tool shall check the database segment directory structure to make sure that it complies with DII COE standards for database segments.

3.2.2.1.1.2.2.1 The data base segment verification tool shall check that the DBS files and install directories are present in the segment's primary subdirectory.

3.2.2.1.1.2.2.2 The data base segment verification tool shall check that the `Scripts` subdirectory contains the `.cshrc` file.

3.2.2.1.1.2.2.3 The data base segment verification tool shall check that the `SegDescrip` subdirectory contains the `PostInstall` file.

3.2.2.1.1.2.3 The data base segment verification tool shall check that the `DBS_files` directory is owned by the Relational Data Base Management System (RDBMS) and not the database segment.

3.2.2.1.1.2.4 The data base segment verification tool shall check that the `SegInfo` descriptor file contains the "`Database`" and "`Requires`" segment descriptor sections.

3.2.2.1.1.2.5 The data base segment verification tool shall check that the "`Database`" descriptor section contains the "`$SCOPE`" descriptor.

3.2.2.1.1.2.6 The data base segment verification tool shall check that if the databases segment being verified modifies another database segment's schema that it has a file to restore those objects when the modified database segment is installed.

3.2.2.1.1.2.7 The data base segment verification tool shall create an output file named "`VSDBOutput`" in the event that no errors and at lease one warning are found. This file shall be placed in the database segment's `SegDescrip` directory.

3.2.2.1.1.2.8 The data base segment verification tool shall update the "`Validated`" file in the `SegDescrip` directory if it was already created by the *COE verify segment* tool and if no errors are found by the *COE verify segment* database tool. The *COE verify segment* database tool will never create the "`Validated`" file, because its absence means that *COE verify segment* tool found errors in the database segment.

3.2.2.1.1.3 The database segment verification tool shall list any inter-database object dependencies that exist over multiple database segments on a common database server.

3.2.2.1.1.3.1 The database segment verification tool shall list any table dependencies by triggers and views.

3.2.2.1.1.3.2 The database segment verification tool shall list any procedure dependencies by triggers.

3.2.2.1.1.3.3 The database segment verification tool shall list any candidate key dependencies by foreign keys.

3.2.2.1.2 Compliance Testing Tools

3.2.2.1.2.1 The developer's tools shall provide a *COE can install* tool with the capability to test a segment to determine if it can be installed.

3.2.2.1.2.1.1 The tool shall check to see if all required segments are already on the system's disk.

3.2.2.1.2.1.2 The tool shall check to ensure there aren't any conflicting segment on the system's disk.

3.2.2.1.2.2 The developer's tools shall provide a *COE check compliance* tool with the capability to examine a segment to determine if is at Category 1 compliance level.

3.2.2.1.2.2.1 The shall report the maximum possible level of compliance based on the criteria that can be checked automatically.

3.2.2.1.2.3 The developer's tools shall provide a *COE verify segment* tool with the capability to validate that a segment conforms to the rules for defining a segment.

3.2.2.1.2.3.1 The tool shall use the information in the `SegDescrip` subdirectory.

3.2.2.1.2.3.2 The tool shall executed whenever a segment is modified.

3.2.2.1.2.3.3 The tool shall be executed against every segment.

3.2.2.1.2.3.4 If the segment is part of an aggregate, the tool shall be executed on each segment in the aggregate.

3.2.2.1.2.3.5 The tool shall check to ensure that executables are prefaced with the segment prefix where required.

3.2.2.1.2.3.6 The tool shall check to ensure that all defined environment variables use the segment prefix.

3.2.2.1.2.3.7 The tool shall check to ensure that all pathnames are defined relative to the home environment variable, `segprefix_HOME`.

3.2.2.1.2.3.8 The tool shall check to ensure that all required environment variable (`USER_HOME`, `USER_DATA`, `DATA_DIR`, etc.) are defined by account group segments.

3.2.2.1.2.3.9 The tool shall check to ensure that all required environment variables are defined by the COE parent component segment.

3.2.2.1.2.3.10 The tool shall check aggregate segments for internal consistency (e.g., all components are hardware compatible).

3.2.2.1.2.3.11 The tool shall check and warn if `mv` is used in a `PostInstall`, `PreInstall`, or `DEINSTALL` script since this may be an illegal attempt to move files across disk partitions.

3.2.2.1.2.4 The developer's tools shall provide a *COE verify COE* tool with the capability to validate the COE runtime environment.

3.2.2.1.2.4.1 The tool shall check for sufficient swap space on the disk.

3.2.2.1.2.4.2 The tool shall check for correct disk partitioning.

3.2.2.1.2.4.3 The tool shall check for expected device drivers.

3.2.2.1.2.4.4 The tool shall check for sufficient shared memory, message pool size, and other operating system kernel parameters.

3.2.2.1.2.4.5 The tool shall check for proper location of X and Motif libraries.

3.2.2.1.3 Segment Submission Tools

3.2.2.1.3.1 The developer's tools shall provide a *COE configuration definition* tool with the capability to create a configuration definition from a list of segments.

3.2.2.1.3.1.1 Components of a configuration definition (e.g., bundles, configuration, folders) shall be able to participate in more than one configuration definition.

3.2.2.1.3.1.2 When segment reside in a software repository, the tool shall provide an interface that allows segments to be checked out of a repository such as the Software Distribution Management System (SDMS).

3.2.2.1.3.1.3 The *COE configuration definition* tool shall have the same interface as the *COE make installation* tool.

3.2.2.1.3.2 The developer's tools shall provide a *COE make installation* tool with the capability to write one or more segments to an installation medium.

3.2.2.1.3.2.1 The tool shall provide the capability to package one or more segments for distribution over SIPRNET.

3.2.2.1.3.2.2 The tool shall check to see if the *COE verify segment* tool has been run successfully on each of the segments.

3.2.2.1.3.2.3 If the *COE verify segment* tool has not been run successfully on each of the segments, the *COE make installation* tool shall abort with an error.

3.2.2.1.3.2.4 The tool shall support multi-volume tapes.

3.2.2.1.3.2.5 The tool shall allow the splitting of segments across tapes. The tape write and read functions shall be able to process segments as if they were all on one tape.

3.2.2.1.3.2.6 The tool shall allow segments for different hardware platforms to be on the same installation medium.

3.2.2.1.3.2.7 The tool shall allow configuration definitions to be written to the installation medium.

3.2.2.1.3.2.8 The tool shall allow command-line parameters and segment lists to be repeated as required.

3.2.2.1.3.2.9 When the tool is processing a segment aggregate, the tool shall accept the aggregate's segment components in any order.

3.2.2.1.3.2.10 When the tool is processing a segment aggregate, the tool shall order the components in such a way as to guarantee that the tape will not need to be rewound during installation.

3.2.2.1.3.2.11 The tool shall require that segments be available on disk with at least the `SegDescrip` subdirectories in an uncompressed, unencrypted format.

3.2.2.1.3.2.12 Since segments are normally installed in a repository such as SDMS, the tool shall provide an interface to allow segments to be checked out from the repository as required.

3.2.2.1.3.2.13 The tool's interface for checking segments out of a repository shall be provided by invoking a user supplied program and passing it the name of the segment to extract and the subdirectory where to place the segment.

3.2.2.1.3.2.14 The tool shall automatically delete temporary segments when finished with them.

3.2.2.1.3.3 The developer's tools shall provide *a COE configuration management make distribution* tool to be used by Configuration Management departments.

3.2.2.1.3.3.1 The tool shall be configurable that allows groups to be defined (e.g., operational sites, developers).

3.2.2.1.3.3.2 The tool shall package all or portions of a submitted segment for delivery to the selected group.

3.2.2.1.3.3.3 The tool shall extract the necessary directories out of the repository and provide them to the *COE make installation* tool for creating the actual distribution media..

3.2.2.1.3.4 The developer's tools shall provide a *COE make submit TAR* tool with the capability to package segments by compressing and encrypting them prior to submitting them electronically to the Software Distribution Management System (SDMS).

3.2.2.1.3.4.1 The tool shall check to see if the COE verify segment tool has been run successfully on the segment.

3.2.2.1.3.4.2 The tool shall check to see if the COE verify segment tool has been run successfully on the segment and aborts with an error is it has not.

3.2.2.1.3.4.3 The tool shall check the directories defined in Chapter 5 of the DII COE I&RTS required for segment submission, are provided.

3.2.2.1.3.4.4 The tool shall allow segments to be check out of a repository prior to packaging for submission.

3.2.2.1.3.4.5 The tool shall package only one segment into a tar file.

3.2.2.1.3.4.6 The tool shall package separately each component of an aggregate.

3.2.2.1.3.5 The developer's tools shall provide a *COE submit* tool with the capability to electronically transmit segments packaged by the *COE make submit TAR* to the host repository.

3.2.2.1.3.5.1 The tool shall allow multiple files to be sent during one session.

3.2.2.1.3.5.2 By default, the tool shall not delete segments from the submitting machine after submission.

3.2.2.1.3.6 The developer's tools shall provide a *COE unpack submit tar* tool with the capability to decompress and decrypt segments that have been packaged with the *COE make submit tar* tool.

3.2.2.1.4 Segmentation Support Tools

3.2.2.1.4.1 The developer's tools shall provide a *COE calculate space* tool with the capability to compute the space required for the segment specified and update the hardware segment descriptor accordingly.

3.2.2.1.4.1.1 The tool shall accept the `-v` (verbose) flag and print out the space requirements for each top-level subdirectory.

3.2.2.1.4.1.2 The tool shall enable the warning flag by default.

3.2.2.1.4.1.3 The tool shall print a warning message if expected directories are missing for the segment type (e.g., `bin` for software segments).

3.2.2.1.4.1.4 The tool shall print a warning message if directories are found that are not expected (e.g., include, `src`).

3.2.2.1.4.1.5 The tool shall not affect the reserve space request in the `Hardware` descriptor.

3.2.2.1.4.1.6 The tool shall not affect any partitions specified.

3.2.2.1.4.1.7 The tool shall accept a reserve value and place it in the appropriate location in the Hardware section of the `SegInfo` file.

3.2.2.1.4.2 The developer's tools shall provide a *COE integrated development environment* tool with a GUI interface for accessing all of the developer tools and any runtime tools that are useful during the development process.

3.2.2.1.4.2.1 The tool shall be used only with COE tools that operate on segments such as the *COE verify segment* tool.

3.2.2.1.4.2.2 The tool shall not be used with COE tools invoked by segments themselves such as the *COE message* tool.

3.2.2.1.4.2.3 The tool shall provide a menu-and-icon driven interface that allows the developer to open segments for development work.

3.2.2.1.4.2.4 Once the GUI is opened, the tool shall provide access to any of the development tools with which to operate on segments.

3.2.2.1.4.2.5 The tool shall allow the output from any of the tools to be captured into a log file for later printing or review.

3.2.2.1.4.2.6 The tool shall provide the capability to set tool flags such as the verbose (`-v`) and warning (`-w`) flags.

3.2.2.1.4.3 The developer's tools shall provide a *COE convert segment* tool with the capability to examine segment descriptors and convert them to the latest format.

3.2.2.1.4.3.1 The tool shall provide, to the extent possible, the capability to identify and correct obsolete usage as part of this process.

3.2.2.1.4.3.2 The tool shall not modify the original segment descriptor directory.

3.2.2.1.4.3.3 The tool shall rename the original segment descriptor directory to `SegDescrip.orig`.

3.2.2.1.4.3.4 The tool shall create converted segment descriptors and place them in a newly created `SegDescrip` directory.

3.2.2.1.4.3.5 To avoid inadvertently overwriting a segment descriptor directory, the tool shall provide a prompt if the directory `SegDescrip.orig` already exists.

3.2.2.1.4.3.6 The tool shall be used to combine individual segment descriptor files into the proper `SegInfo` descriptor file.

3.2.2.1.4.3.7 The tool shall print a warning if obsolete directories, such as `progs` and `libs`, are encountered.

3.2.2.1.4.4 The developer's tools shall provide a *COE make attributes* tool with the capability to create the descriptor file `FileAttribs`.

3.2.2.1.4.4.1 The tool shall recursively traverse every subdirectory beneath the segment home directory and creates a file with lines in the format:

```
permits:owner:group:filename
```

3.2.2.1.4.4.2 At installation time, `FileAttribs` shall be used by installation tools to perform the following statements for each entry:

```
chmod permits $INSTALL_DIR/filename
chown owner $INSTALL_DIR/filename
chgrp group $INSTALL_DIR/filename
```

3.2.2.1.4.4.3 For security reasons, the tool shall not include any file owned by `root` nor any file for which `permits` is greater than 777.

3.2.2.1.4.4.4 The tool shall print a warning for each filename that is rejected.

3.2.2.1.4.4.5 The tool shall print a warning for each “suspicious” permission setting such as 777 for a file.

3.2.2.1.4.4.6 The tool shall print a warning for execute permissions on files in a `data` subdirectory.

3.2.2.1.4.5 The developer’s tools shall provide a *COE move segment* tool with the capability to allow a segment that has already been installed to be moved from one location to another location on the same disk.

3.2.2.1.4.5.1 The tool shall provide the capability to allow a segment that has already been installed to be moved from one disk partition location to a different disk partition location.

3.2.2.1.4.5.2 The tool shall generate an error if there is not enough room to move the segment.

3.2.2.1.4.5.3 The tool shall delete the old segment location when the move is completed.

3.2.2.1.4.6 The developer’s tools shall provide a *COE segment analyze* tool with the capability to analyze the segment specified and create a table showing total memory and disk usage.

3.2.2.1.4.6.1 The tool shall determine all segment dependencies so that the statistics reported include all required segments.

3.2.2.1.4.7 The developer’s tools shall provide a *COE segment catalog* tool with the capability to add a segment to the segment catalog.

3.2.2.1.4.8 The developer’s tools shall provide a *COE test install* tool with the capability to temporarily install a segment that already resides on disk.

3.2.2.1.4.8.1 The tool shall be executed when there are no other COE processes running.

3.2.2.1.4.8.2 The tool shall provide the capability to locate the segment in any arbitrary location to test install the segment.

3.2.2.1.4.8.3 The tool shall establish the required symbolic link under `/h` to preserve the COE standard directory structure.

3.2.2.1.4.9 The developer’s tools shall provide a *COE test remove* tool with the capability to remove a segment that was installed by the *COE test install* tool.

3.2.2.1.4.9.1 The tool shall be executed when there are no other COE processes running.

3.2.2.1.4.9.2 The tool shall remove the symbolic link under `/h` if one exists.

3.2.2.1.4.9.3 The tool shall not delete the segment from disk.

3.2.2.1.4.10 The developer's tools shall provide a *COE time stamp* tool with the capability to put the current time and date into the `VERSION` segment descriptor.

3.2.2.1.4.10.1 The tool shall provide the capability to update the time stamp just prior to running the *COE verify segment* tool and the *COE make submit TAR* tool for the deliverable product.

3.2.2.1.4.11 The developer's tools shall provide a *COE verify update* tool with the capability to update the segment version number, date, and time in the `VERSION` descriptor.

3.2.2.1.4.11.1 The tool shall provide command line parameters to indicate which digits (e.g., major release, minor release, maintenance digit, or developer digit) to update.

3.2.2.1.4.11.2 The tool shall allow multiple digits to be updated at one time.

3.2.2.1.4.11.3 The tool shall, by default, update only the maintenance digit.

3.2.2.1.4.11.4 If no version number is specified, the tool shall increment the version number inside the `VERSION` descriptor; that is, `1.0.0.0` is updated to be `1.0.0.1`.

3.2.2.1.4.11.5 If no version number is specified, and `VERSION` does not exist or has no version number, the tool shall create the `VERSION` descriptor and version number `1.0.0.0` inserted.

3.2.2.1.4.11.6 The tool shall not perform any error checking on the version number specified since the segment must still be validated by the *COE verify segment* tool.

3.2.2.1.4.12 The developer's tools shall provide a *COE verify security* tool with the capability to perform security checks against a segment, against an installed COE, or against an installed system.

3.2.2.1.4.12.1 The tool shall provide the capability to perform security checks against an installed COE.

3.2.2.1.4.12.2 The tool shall provide the capability to perform security checks against an installed system.

3.2.2.1.4.12.3 The tool shall identify potential security risks that are common across all system (e.g., files that are world readable/writeable, use of remote commands, improper `.rhost` files, etc.).

3.2.2.1.4.12.4 When used against an entire system, the tool shall provide interfaces to commercially available tools such as SATAN to identify vulnerability areas.

3.2.2.1.4.12.5 The tool shall perform virus checking by providing an interface to commercially available virus checkers.

3.2.2.1.4.13 The developer's tools shall provide a *Distributed Computing Environment (DCE)* tool with the capability to assist in the development of DCE-base application segments.

3.2.2.1.4.13.1 The tool shall include example client and server software templates that demonstrate typical usage of the DCE capabilities for each of the supported programming languages.

3.2.2.1.4.13.2 The tool shall provide a template or example of DCE wrappers that show how to encapsulate an existing non-DCE executable application such that it can be invoked via DCE.

3.2.2.1.4.13.3 The tool shall provide a template or example of DCE wrapper backends that perform command line-based service invocation.

3.2.2.2 API's

To Be Supplied (TBS)

3.3 CSCI External Interface Requirements

The COE tools can be accessed in two different ways. First, a user program can be written and linked with the tools library. Such a program goes through the defined APIs to request services from the tools. A second way that the tools can be used is to execute the tools as a separate program and use services provided by the operating system to communicate between the tool and the calling program. This technique is most frequently used as a way to invoke the tools from within a PostInstall script. Whichever approach is most convenient may be used. All of the tools are available as individual executables, but not all are available through a library interface. When accessed through a library interface, the tools return data through the interface. They may also write to stdout. When used as separate executable programs, the tools communicate with the outside world in two ways.

The tools use the exit function to set the UNIX status environment variable. (status is used for the C shell csh. The **$?** environment variable is used for POSIX shells.) status is set to 0 for normal tool completion. A status return value other than 0 indicates a completion code that is tool-specific. Refer to the appropriate Developer's Toolkit documentation on man pages for a complete description of the tools and their return codes, including error return codes.

The tools use stdin, stdout, and stderr and thus support I/O redirection. Redirecting stdin allows the tools to receive input from a file or another program, while redirecting stdout allows the tools to provide input to other programs. (Redirecting stdin is not always convenient. The -R command-line parameter, allows a tool to read

input from a response file instead ofstdin.) The tools write normal output to stdout, while errors and warnings are sent to stderr.

3.3.1 Interface Identification and Diagrams

3.3.x (Project-unique identifier of interface)

3.4 CSCI internal interface requirements

3.5 CSCI internal data requirements

3.6 Adaptation requirements

3.7 Safety requirements

3.8 Security and privacy requirements

3.9 CSCI environment requirements

3.10 Computer resource requirements

3.10.1 Computer hardware requirements

3.10.2 Computer hardware resource utilization requirements

3.10.3 Computer software requirements

3.10.4 Computer communications requirements

3.11 Software quality factors

3.12 Design and implementation constraints

3.13 Personnel-related requirements

3.14 Training-related requirements

3.15 Logistics-related requirements

3.16 Other requirements

3.17 Packaging requirements

3.18 Precedence and criticality of requirements

4. Qualification Provisions

5. Requirements Traceability

This section in its entirety will be removed from the document prior to posting on the DII COE HomePage or any other web site. Please reference the distribution statements on the cover page of this document for further guidance.

5.1 Objectives of Traceability

Traceability, in the context of the DII COE, means that each of the requirements identified in section 3.2, Toolkit (TK) Functional Area Capability Requirements, of this document is associated with a non-segmented software tools and libraries that satisfies the requirement. These toolkit components that satisfy the requirement may differ by the DII COE Version, the supported operating system, or both. The matrix in section 5.2 will identify this level of detail.

5.2 Requirements Traceability Matrix (RTM)

For security purposes, the Toolkit SRS RTM will be available as a separate document.

6. Notes

A. Appendixes

Change Control Page

Date	Version No.	Reason for Version Change
16 March 1997	V1.0	Reformat and Clean up
14 April 1997	V1.1	Initial Review by Toolkit (TK) TWG and acceptance of changes agreed to during the Review. 1. TKTWG approval of V1.1 as an initial baseline. 2. Section 1.2, 2nd Para., 1st sentence: Deleted the word "COE" and replaced it with "repository". 3. Section 1.2, 2nd Para., 2nd sentence: Deleted the word "COE". 4. Section 1.2, last Para., 1st sentence: Deleted the phrase "User System Interface Style Guide". 5. Section 3.2.2.1.4.2 Deleted the phrase "include the capability to provide a GUI for" and replaced it with "provide an interface for".
20 May 1997	V1.2	1. Added proposed changes submitted by Tom Miller, NOSC. 2. Added detailed requirements from I&RTS 3. Added 2 missed requirements from I&RTS. 4. Added Section 5, Requirements Traceability. 5. Global change "shall include" to "shall provide". 6. Revised Section 2, Referenced Documents.
26 June 1997	V1.3	1. Revised Sections 1.1, 1.2, and 1.3. Took out specific references to DII COE version numbers, corrected references to Toolkit TWG and DII COE Toolkit SRS. 2. Revised Section 3.2 intoto. Removed references to JAW and NOSC. 3. Revised Section 5 Added text to title paragraph, and added Sections 5.1 and 5.2 with supporting text. Requirements Traceability Matrix (RTM) is now in Section 5.2. 4. Revised new Section 5.2 Repaginated and removed references to JAW and NOSC from requirements text as similarly done for Section 3.2.

SRS Para	I&RTS Tool Name	I&RTS Section Ref	Tool Exists
3.2.1.1	**User Interface**		
3.2.1.1.1	COEPrompt	C-2.31	√
3.2.1.1.2	COEAskUser	C-2.2	√
3.2.1.1.3	COEMsg	C-2.30	√
3.2.1.1.4	COEPromptPasswd	C-2.32	√
3.2.1.2	**Database Tools**		
3.2.1.2.1	COEAddDBUser	C-2.1	
3.2.1.2.2	COEDeleteDBUser	C-2.9	
3.2.1.2.3	COEBackupDB	C-2.3	
3.2.1.2.4	COELstDBDepends	C-2.21	
3.2.1.2.5	COELstDBRoleUsrs	C-2.22	
3.2.1.2.6	COELstUsrDBRoles	C-2.27	
3.2.1.2.7	COERestoreDB	C-2.37	
3.2.1.2.8	COEStartDBServer	C-2.40	
3.2.1.2.9	COEStopDBServer	C-2.41	
3.2.1.2.10	COECreateDS	C-2.8	
3.2.1.2.11	COEDropDS	C-2.11	
3.2.1.2.12	COEExtendDS	C-2.12	
3.2.1.2.13	COERebuildIndex	C-2.34	
3.2.1.2.14	COERebuildView	C-2.35	
3.2.1.3	**System Administration Tools**		
3.2.1.3.1	COEChkUpdates	C-2.4	
3.2.1.3.2	COEConnectAS	C-2.5	
3.2.1.3.3	COECopyWS	C-2.6	
3.2.1.3.4	COECreateAS	C-2.7	
3.2.1.3.5	COEDelUnused	C-2.10	
3.2.1.3.6	COEFindData	C-2.13	
3.2.1.3.7	COEFindServer	C-2.16	
3.2.1.3.8	COEListSegs	C-2.20	
3.2.1.3.9	COEListDepends	C-2.19	
3.2.1.3.10	COEPropagateUpdates	C-2.33	
3.2.1.3.11	COERemoveAS	C-2.36	
3.2.1.3.12	COETestInstall	C-2.42	√
3.2.1.4	**Profiling Tools**		
3.2.1.4.1	COELstProfileSegs	C-2.23	
3.2.1.4.2	COELstProfileUsrs	C-2.24	
3.2.1.4.3	COELstSegProfiles	C-2.25	

SRS Para	I&RTS Tool Name	I&RTS Section Ref	Tool Exists
3.2.1.4.4	COELstSegUsrs	C-2.26	
3.2.1.4.5	COELstUsrProfiles	C-2.28	
3.2.1.4.6	COELstUsrSegs	C-2.29	
3.2.1.5	**Installation Tools**		
3.2.1.5.1	COEInstaller	C-2.17	√
3.2.1.5.2	COEInstError	C-2.18	√
3.2.1.5.3	COERmtInstall	C-2.38	
3.2.1.5.4	COEScanCOTS	C-2.39	
3.2.1.5.5	COETestRemove	C-2.43	√
3.2.1.5.6	COEUpdateHome	C-2.44	√
3.2.1.5.7	COEFindDrive	C-2.14	
3.2.1.5.8	COEFindSeg	C-2.15	
3.2.2.1.1	**Database Tools**		
3.2.2.1.1.1	ChkDBCompliance	C-3.4	
3.2.2.1.1.2	VerifySegDB	C-3.23	
3.2.2.1.1.2.1	[NEW]		
3.2.2.1.1.2.1.1	[NEW]		
3.2.2.1.1.2.1.2	[NEW]		
3.2.2.1.1.2.1.3	[NEW]		
3.2.2.1.1.2.2	[NEW]		
3.2.2.1.1.2.3	[NEW]		
3.2.2.1.1.2.4	[NEW]		
3.2.2.1.1.2.5	[NEW]		
3.2.2.1.1.2.6	[NEW]		
3.2.2.1.1.2.7	[NEW]		
3.2.2.1.1.3	[NEW]		
3.2.2.1.1.3.1	[NEW]		
3.2.2.1.1.3.2	[NEW]		
3.2.2.1.1.3.3	[NEW]		
3.2.2.1.2	**Compliance Testing Tools**		
3.2.2.1.2.1	CanInstall	C-3.2	√
3.2.2.1.2.2	ChkCompliance	C-3.3	
3.2.2.1.2.3	VerifySeg	C-3.22	√
3.2.2.1.2.4	VerifyCOE	C-3.20	
3.2.2.1.3	**Segment Submission Tools**		
3.2.2.1.3.1	ConfigDef	C-3.7	

SRS Para	I&RTS Tool Name	I&RTS Section Ref	Tool Exists
3.2.2.1.3.2	MakeInstall	C-3-10	√
3.2.2.1.3.3	CMMakeDistrib	C-3.5	
3.2.2.1.3.4	mkSubmitTar	C-3.11	
3.2.2.1.3.5	submit	C-3.15	
3.2.2.1.3.6	unpackSubmitTar	C-3.19	
3.2.2.1.4	**Segmentation Support Tools**		
3.2.2.1.4.1	CalcSpace	C-3.1	√
3.2.2.1.4.2	COEIDE	C-3.6	
3.2.2.1.4.3	ConvertSeg	C-3.8	
3.2.2.1.4.4	MakeAttribs	C-3.9	√
3.2.2.1.4.5	MoveSeg	C-3.12	
3.2.2.1.4.6	SegAnalyze	C-3.13	
3.2.2.1.4.7	SegCatalog	C-3.14	
3.2.2.1.4.8	TestInstall	C-3.16	√
3.2.2.1.4.9	TestRemove	C-3.17	√
3.2.2.1.4.10	TimeStamp	C-3.18	√
3.2.2.1.4.11	VerUpdate	C-3.24	√
3.2.2.1.4.12	VerifySecurity	C-3.21	